Simulation Code Writeup
First Quarter 2009-2010

Jeff Hobson

October 30, 2009

1 Environment

1.1 Code

```
File: ALMSim.pde
final int WIDTH     = 400;
final int HEIGHT    = WIDTH;
final int MARGIN    = 50;
int[] stdGrassColor = { 50, 160, 50};
int[] cutGrassColor = { 50, 255, 50};
int[] groundColor   = {115, 50, 0};
int[] obstacleColor = {255, 75, 75};
int[][] floorType   = new int[HEIGHT][WIDTH]; // 0 = uncut; 1 = cut; 2 = gnd; 3 = obs; 4
Robot ALM;
void setup()
{
  size(500,500);
  stroke(0);
  fill(255);
  ALM = new Robot();
  //hard-coded or randomly-generated map
  //cut
  for(int r=25; r<75; r++)
    for(int c=25; c<75; c++)
      floorType[r][c] = 1;
  //ground
  for(int r=100; r<125; r++)
    for(int c=25; c<75; c++)
      floorType[r][c] = 2;
  //obstacle(s)
  for(int r=150; r<175; r++)
    for(int c=25; c<75; c++)
      floorType[r][c] = 3;
```

```
  //noLoop();
  //startSim();
}
void draw()
{
  //cut
  ALM.cutForward();
  //borders
  noFill();
  stroke(0);
  rectMode(LEFT);
  rect(MARGIN-1,MARGIN-1,WIDTH+MARGIN,HEIGHT+MARGIN);
  //map
  for(int r=0; r<HEIGHT; r++)
  {
    for(int c=0; c<WIDTH; c++)
    {
      if(floorType[r][c] == 0)
      {
stroke(stdGrassColor[0], stdGrassColor[1], stdGrassColor[2]);
point(MARGIN+c,MARGIN+r);
      }
      else if(floorType[r][c] == 1)
      {
stroke(cutGrassColor[0], cutGrassColor[1], cutGrassColor[2]);
point(MARGIN+c,MARGIN+r);
      }
      else if(floorType[r][c] == 2)
      {
stroke(groundColor[0], groundColor[1], groundColor[2]);
point(MARGIN+c,MARGIN+r);
      }
      else if(floorType[r][c] == 3)
      {
stroke(obstacleColor[0], obstacleColor[1], obstacleColor[2]);
point(MARGIN+c,MARGIN+r);
      }
      /*else if(floorType[r][c] == 4)
      {
int[] co = ALM.getColor();
stroke(co[0],co[1],co[2]);
point(MARGIN+c,MARGIN+r);
      }*/
      else
      {
fill(255);
```

```
point(MARGIN+c,MARGIN+r);
      }
    }
  //robot
  ALM.Draw();
  }
}
void startSim()
{
  int count = 0;
  while(count<(WIDTH-5)/ALM.getSpeed())
  {
    ALM.cutForward();
    redraw();
    count++;
  }
}
```

1.2 Discussion

This code is creates the environment for the robot (the lawn). It is created manually (hard-coded) for now, but it will eventually have the capability to create random lawns with obstacles of different sizes and shapes. The ALMSim class instantiates the Robot object in the `setup()` method with the default values. The `draw()` method is a method that is required in every Processing base class and is looped indefinitely by the program until the user decides to exit. This method will update the lawn based on the value of each pixel and also calls the Robot's `Draw()` method, which prints the lawnmower to the screen.

2 Object

2.1 Code

```
File: Robot.pde
class Robot
{
  private final double left      = 90;
  private final double right     = 270;
  private final double halfLeft = 45;
  private final double halfRight = 225;
  private int row, col;   //current location on grid
  private int speed;      //pixels per second
  private int roboSize;   //size in pixels
  private int viewRange; //distance sensors can reach in pixels
  private double heading; //direction in polar grid
```

```
private String type;    //circular or rectangular size
private int[] roboColor = {0,0,220};
//Constructors
public Robot(int r, int c, int sp, double h, int sz, int vr, String t, int[] cl)
{
  row       = r;
  col       = c;
  speed     = sp;
  heading   = h;
  roboSize = sz;
  viewRange = vr;
  type      = t;
  roboColor = cl;
}
public Robot(int r, int c, int sp, double h, int sz, int vr)
{
  row       = r;
  col       = c;
  speed     = sp;
  heading   = h;
  roboSize = sz;
  viewRange = vr;
  type = "rect";
}
public Robot(int r, int c, int sp, double h)
{
  row       = r;
  col       = c;
  speed     = sp;
  heading   = h;
  roboSize = 5;
  viewRange = 3;
  type = "rect";
}
public Robot()
{
  row       = HEIGHT-6;
  col       = 5+40;
  speed     = 20;
  heading   = 1;
  roboSize = 5;
  viewRange = 3;
  type = "rect";
}
//Accessors
int[] getPos()
```

```
{
  int[] ret = {row, col};
  return ret;
}
int getSpeed()
{
  return speed;
}
double getHeading()
{
   return heading;
}
int getSize()
{
   return roboSize;
}
int getRange()
{
   return viewRange;
}
String getType()
{
   return type;
}
int[] getColor()
{
   return roboColor;
}
//"Setters"
void setPos(int r, int c)
{
   row = r;
   col = c;
}
void setSpeed(int sp)
{
   speed = sp;
}
void setHeading(int h)
{
   heading = h;
}
void setSize(int sz)
{
   roboSize = sz;
}
```

```
 void setRange(int vr)
 {
    viewRange = vr;
 }
 void setType(String t)
 {
    type = t;
 }
 void setColor(int[] c)
 {
    roboColor = c;
 }
 //Instance Methods
 void Draw()
 {
    /*for(int r=-roboSize; r<roboSize; r++)
      for(int c=-roboSize; c<roboSize; c++)
if(row+r<0 || col+c<0 || row+r>=HEIGHT || col+c>=WIDTH)
   continue;
else
   floorType[row+r][col+c] = 4;*/
    rectMode(CENTER);
    stroke(roboColor[0],roboColor[1],roboColor[2]);
    fill(roboColor[0],roboColor[1],roboColor[2]);
    rect(col+MARGIN,row+MARGIN,roboSize*2,roboSize*2);
    //noStroke();
    //noFill();
 }
 void cutForward()
 {
    boolean flag = false;
    for(int x=0; x<speed; x++)
    {
      for(int r=-roboSize; r<=roboSize; r++)
 for(int c=-roboSize; c<=roboSize; c++)
   if(row+r<0 || col+c<0 || row+r>=HEIGHT || col+c>=WIDTH || floorType[row+r][col+c]==3
   {
     flag = true;
     if(row+r<0) row = -r;
     if(col+c<0) col = -c;
     if(row+r>=HEIGHT) row = HEIGHT-r-1;
     if(col+c>=WIDTH) col = WIDTH-c-1;
     if(floorType[row+r][col]==3) row = row-(r<1?-1:1);
     if(floorType[row][col+c]==3) col = col-(c<1?-1:1);
     continue;
   }
```

```
   else
     floorType[row+r][col+c] = 1;
      if(!flag)
      {
int[] pos = forward(col, row); //move to new position
col = pos[0];
row = pos[1];
      }
      else
      {
 turn(left);
 break;
      }
    }
    //draw robot again
    //this.Draw();
 }
 int[] forward(int cola, int rowb)
 {
    int c = cola;
    int r = rowb;
    if(heading == 0)         //EAST
      c++;
    else if(heading == 1)    //NORTH
      r--;
    else if(heading == 2)    //WEST
      c--;
    else if(heading == 3)    //SOUTH
      r++;
    else if(heading == 0.5) //NORTHEAST
    {
      c++;
      r--;
    }
    else if(heading == 1.5) //NORTHWEST
    {
      c--;
      r--;
    }
    else if(heading == 2.5) //SOUTHWEST
    {
      c--;
      r++;
    }
    else if(heading == 3.5) //SOUTHEAST
    {
```

```
      c++;
      r++;
    }
    else
      print("Error!!!"); //Soon to be changed.
    int[] retval = {c,r};
    return retval;
  }
  void turn(double theta)
  {
    if(theta == left)
      heading++;
    else if(theta == right)
      heading--;
    else if(theta == halfLeft)
      heading += .5;
    else if(theta == halfRight)
      heading -= .5;
    if(heading >= 4)
      heading -= 4;
    if(heading < 0)
      heading += 4;
    println(heading);
  }
}
```

2.2 Discussion

The Robot class is a private class included within the ALMSim class. The Robot contains various methods used to access and set its private variables. The class also has four methods, `Draw, turn, cutForward,` and `forward,` that aid in the moving process. Before moving forward, the Robot will check to see if it is at a boundary or an obstacle. If so, it turns left, otherwise, it will continue in the current heading. The Robot class allows for movement in 8 directions: N, NW, W, SW, S, SE, E, and NE. Whenever the Robot moves forward, it replaces the pixel values at where it was to "1", the value for cut grass. The next step for the Robot class is to have the capacity to move over "ground" (pixels with value "2") without leaving the value for cut grass behind.